Assignment 01 Code and Outputs

Assignment 1.1 (Part a) – Run Keras MNIST MLP Example

Code:

*'''Trains a simple deep NN on the MNIST dataset.  
  
Gets to 98.40% test accuracy after 20 epochs  
(there is \*a lot\* of margin for parameter tuning).  
2 seconds per epoch on a K520 GPU.  
'''*from tensorflow import keras  
from tensorflow.keras.datasets import mnist  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense, Dropout  
from tensorflow.keras.optimizers import RMSprop  
  
batch\_size = 128  
num\_classes = 10  
epochs = 20  
  
# the data, split between train and test sets  
(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()  
  
x\_train = x\_train.reshape(60000, 784)  
x\_test = x\_test.reshape(10000, 784)  
x\_train = x\_train.astype('float32')  
x\_test = x\_test.astype('float32')  
x\_train /= 255  
x\_test /= 255  
print(x\_train.shape[0], 'train samples')  
print(x\_test.shape[0], 'test samples')  
  
# convert class vectors to binary class matrices  
y\_train = keras.utils.to\_categorical(y\_train, num\_classes)  
y\_test = keras.utils.to\_categorical(y\_test, num\_classes)  
  
model = Sequential()  
model.add(Dense(512, activation='relu', input\_shape=(784,)))  
model.add(Dropout(0.2))  
model.add(Dense(512, activation='relu'))  
model.add(Dropout(0.2))  
model.add(Dense(num\_classes, activation='softmax'))  
  
model.summary()  
  
model.compile(loss='categorical\_crossentropy',  
 optimizer=RMSprop(),  
 metrics=['accuracy'])  
  
history = model.fit(x\_train, y\_train,  
 batch\_size=batch\_size,  
 epochs=epochs,  
 verbose=1,  
 validation\_data=(x\_test, y\_test))  
score = model.evaluate(x\_test, y\_test, verbose=0)  
print('Test loss:', score[0])  
print('Test accuracy:', score[1])

Output:

60000 train samples

10000 test samples

2023-03-18 12:52:52.363258: I tensorflow/core/platform/cpu\_feature\_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 512) 401920

dropout (Dropout) (None, 512) 0

dense\_1 (Dense) (None, 512) 262656

dropout\_1 (Dropout) (None, 512) 0

dense\_2 (Dense) (None, 10) 5130

=================================================================

Total params: 669,706

Trainable params: 669,706

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch 1/20

469/469 [==============================] - 6s 7ms/step - loss: 0.2546 - accuracy: 0.9217 - val\_loss: 0.1334 - val\_accuracy: 0.9567

Epoch 2/20

469/469 [==============================] - 3s 7ms/step - loss: 0.1048 - accuracy: 0.9676 - val\_loss: 0.0820 - val\_accuracy: 0.9730

Epoch 3/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0764 - accuracy: 0.9766 - val\_loss: 0.0621 - val\_accuracy: 0.9794

Epoch 4/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0596 - accuracy: 0.9811 - val\_loss: 0.0631 - val\_accuracy: 0.9809

Epoch 5/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0497 - accuracy: 0.9844 - val\_loss: 0.0707 - val\_accuracy: 0.9791

Epoch 6/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0400 - accuracy: 0.9871 - val\_loss: 0.0569 - val\_accuracy: 0.9830

Epoch 7/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0343 - accuracy: 0.9890 - val\_loss: 0.0757 - val\_accuracy: 0.9811

Epoch 8/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0298 - accuracy: 0.9908 - val\_loss: 0.0697 - val\_accuracy: 0.9820

Epoch 9/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0251 - accuracy: 0.9916 - val\_loss: 0.0712 - val\_accuracy: 0.9816

Epoch 10/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0233 - accuracy: 0.9927 - val\_loss: 0.0727 - val\_accuracy: 0.9818

Epoch 11/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0212 - accuracy: 0.9928 - val\_loss: 0.0705 - val\_accuracy: 0.9831

Epoch 12/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0191 - accuracy: 0.9937 - val\_loss: 0.0765 - val\_accuracy: 0.9823

Epoch 13/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0166 - accuracy: 0.9946 - val\_loss: 0.0765 - val\_accuracy: 0.9836

Epoch 14/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0155 - accuracy: 0.9951 - val\_loss: 0.0809 - val\_accuracy: 0.9831

Epoch 15/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0137 - accuracy: 0.9955 - val\_loss: 0.0860 - val\_accuracy: 0.9832

Epoch 16/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0123 - accuracy: 0.9959 - val\_loss: 0.0845 - val\_accuracy: 0.9838

Epoch 17/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0109 - accuracy: 0.9966 - val\_loss: 0.0828 - val\_accuracy: 0.9831

Epoch 18/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0131 - accuracy: 0.9960 - val\_loss: 0.0783 - val\_accuracy: 0.9853

Epoch 19/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0089 - accuracy: 0.9971 - val\_loss: 0.0871 - val\_accuracy: 0.9832

Epoch 20/20

469/469 [==============================] - 3s 7ms/step - loss: 0.0093 - accuracy: 0.9972 - val\_loss: 0.0841 - val\_accuracy: 0.9843

Test loss: 0.08412754535675049

Test accuracy: 0.9843000173568726

Process finished with exit code 0

Assignment 1.1 (Part b) – Run PySpark Example

Code:

# Licensed to the Apache Software Foundation (ASF) under one or more  
# contributor license agreements. See the NOTICE file distributed with  
# this work for additional information regarding copyright ownership.  
# The ASF licenses this file to You under the Apache License, Version 2.0  
# (the "License"); you may not use this file except in compliance with  
# the License. You may obtain a copy of the License at  
#  
# http://www.apache.org/licenses/LICENSE-2.0  
#  
# Unless required by applicable law or agreed to in writing, software  
# distributed under the License is distributed on an "AS IS" BASIS,  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
# See the License for the specific language governing permissions and  
# limitations under the License.  
#  
  
import sys  
from random import random  
from operator import add  
  
from pyspark.sql import SparkSession  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 """  
 Usage: pi [partitions]  
 """  
 spark = SparkSession\  
 .builder\  
 .appName("PythonPi")\  
 .getOrCreate()  
  
 partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2  
 n = 100000 \* partitions  
  
 def f(\_):  
 x = random() \* 2 - 1  
 y = random() \* 2 - 1  
 return 1 if x \*\* 2 + y \*\* 2 <= 1 else 0  
  
 count = spark.sparkContext.parallelize(range(1, n + 1), partitions).map(f).reduce(add)  
 print("Pi is roughly %f" % (4.0 \* count / n))  
  
 spark.stop()

Output:

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

23/03/19 07:07:39 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Pi is roughly 3.136960

Process finished with exit code 0

Assignment 1.2 Data Sizes, Scaling, Reliability, and Latency

Code:

---  
title: Assignment 1  
subtitle: Computer performance, reliability, and scalability calculation  
author: Jake Meyer  
---  
  
*## 1.2   
  
#### a. Data Sizes*| Data Item | Size per Item |   
|--------------------------------------------|--------------:|  
| 128 character message. | 128 Bytes |  
| 1024x768 PNG image | 3.15 MB |  
| 1024x768 RAW image | 1.57 MB |   
| HD (1080p) HEVC Video (15 minutes) | 224,004 MB |  
| HD (1080p) Uncompressed Video (15 minutes) | 167,962 MB |  
| 4K UHD HEVC Video (15 minutes) | 5,062.50 MB |  
| 4k UHD Uncompressed Video (15 minutes) | 160,180.66 MB |  
| Human Genome (Uncompressed) | ~200 GB |  
  
Assumptions: <br>  
1. All videos are 30 frames per second.  
2. HEVC stands for High Efficiency Video Coding  
3. See the Wikipedia article on display resolution for information <br>  
on HD (1080p) and 4K UHD resolutions.  
  
Calculations: <br>  
1 byte = 1 letter in computer memory therefore 128 characters\* 1 byte = 128 Bytes <br>  
[Image Size Calculator](*https://toolstud.io/photo/filesize.php?imagewidth=1024&imageheight=768*) <br>  
PNG Image File Uncompressed 4x8bit RGBA <br>  
Raw Image FIle Uncompressed 16bit monochrome <br>  
[Video Size Calculator](*https://www.omnicalculator.com/other/video-size*) <br>  
HD (1080p) Cineon 1080 RGB, 15 minute video, 30 frames per second <br>  
HD (1080p) Uncompressed 1080 8-bit, 15 minute video, 30 frames per second <br>  
[4K Video Size Calculator](*https://www.videoproc.com/edit-4k-video/video-size-calculator.htm*) <br>  
4K UHD HEVC Video Resolution 3840 x 2160 from Wikipedia, 15 minutes, 30 frames per second <br>  
4K UHD HEVC Uncompressed Resolution 1920 x 1080 pixels, 15 minutes, 30 frames per second <br>  
[Human Genome Reference](*https://medium.com/precision-medicine/how-big-is-the-human-genome-e90caa3409b0*) <br>  
  
*#### b. Scaling*| | Size | # HD |   
|-------------------------------------------|---------------------:|-----------:|  
| Daily Twitter Tweets (Uncompressed) | 178.81 GB | 1 |  
| Daily Twitter Tweets (Snappy Compressed) | 3,039.83 GB | 1 |  
| Daily Instagram Photos | 692,138.67 GB | 676 |  
| Daily YouTube Videos | 78,751,406.25 GB | 76,906 |  
| Yearly Twitter Tweets (Uncompressed) | 65,267.08 GB | 64 |  
| Yearly Twitter Tweets (Snappy Compressed) | 1,109,540.46 GB | 1,084 |  
| Yearly Instagram Photos | 252,630,859.38 GB | 246,710 |  
| Yearly YouTube Videos | 28,744,263,281.25 GB | 28,070,570 |  
  
  
Assumptions: <br>  
1. Using the estimates for data sizes in part a.  
2. Using 10 TB Hard Drives and storing the data using the Hadoop Distributed File System (HDFS). 3x the amount of storage required. <br>  
3. 500 million tweets sent each day. Each tweet is 128 characters. Snappy compression ratio assumed 1:1.7.  
4. 100 million videos and photos are uploaded to Instagram daily. 75% of those items are 1024x768PNG photos.  
5. Youtube estimates 500 hours of video uploaded every minute. All videos are HD quality encoded using HEVC at 30 fps. <br>  
  
Calculations: <br>  
Method used for calculations was to find the size initially for each request. <br>  
Ensure the data size is multiplied by 3 since using HDFS. <br>  
Use [Data Size Calculator](*https://www.calculator.com/calculate/data-size/*) to understand amount in Terabytes. <br>  
Daily Twitter Tweets Size = 128 bytes \* 500 million tweets \* 3 <br>  
Daily Twitter Tweets Compressed Size = 128 bytes \* 1.7 Compression Ratio \* 500 million tweets \* 3 <br>  
Daily Instagram Photos Size = 3.15 MB \* 100,000,000 photos \* .75 \* 3 <br>  
Daily Youtube Videos Size = 224,004 MB \* 4 \* 500 hours of video \* 60 minutes \* 3 <br>  
Yearly Twitter Tweets Size = 128 bytes \* 500 million tweets \* 365 days \* 3 <br>  
Yearly Twitter Tweets Compressed Size = 128 bytes \* 1.7 Compression Ratio \* 500 million tweets \* 365 days \* 3 <br>  
Yearly Instagram Photos Size = 3.15 MB \* 100,000,000 photos \* .75 \* 365 days \* 3 <br>  
Yearly Youtube Videos Size = 224,004 MB \* 4 \* 500 hours of video \* 60 minutes \* 365 days\* 3 <br>  
  
*#### c. Reliability*| | # HD | # Failures |  
|------------------------------------|-----------:|-----------:|  
| Twitter Tweets (Uncompressed) | 64 | 1 |  
| Twitter Tweets (Snappy Compressed) | 1,084 | 15 |  
| Instagram Photos | 246,710 | 3,380 |  
| YouTube Videos | 28,080,570 | 384,704 |  
  
Assumptions: <br>  
1. Use the yearly estimates from part b. <br>  
  
Calculations: <br>  
[According to Backblaze hard drive statistics](*https://www.backblaze.com/b2/hard-drive-test-data.html*), <br>   
annual failure rate is 1.37%. Calculations will be: <br>  
Annual Number of Hard Drive Failures = Number of HD \* 0.0137 <br>  
Rounded to the nearest whole hard drive.  
*#### d. Latency*| | One Way Latency |  
|---------------------------|----------------:|  
| Los Angeles to Amsterdam | 70.36 ms |  
| Low Earth Orbit Satellite | 100 ms |  
| Geostationary Satellite | 140 ms |  
| Earth to the Moon | 1,280 ms |  
| Earth to Mars | 20 minutes |   
  
Calculations: <br>  
[Global Ping Statistics](*https://wondernetwork.com/pings*) used for latency from Los Angeles to Amsterdam <br>  
Los Angeles to Amsterdam = 140.72 ms / 2 <br>  
[Low to Medium Earth Orbit Satellite](*https://www.satellitetoday.com/telecom/2009/09/01/minimizing-latency-in-satellite-networks/*) <br>  
Low to Medium Earth Orbit Satellite = 100 ms taken from the article. <br>  
[Geostationary Satellite Latency](*https://www.satsig.net/latency.htm*) <br>  
Geostationary Satellite = 280 ms / 2 <br>  
[Earth to Moon and Earth to Mars](*https://short-informer.com/what-is-the-latency-between-earth-and-moon/#:~:text=What%20is%20the%20latency%20between%20earth%20and%20moon%3F,to%20about%205.8%20milliseconds%20of%20wave%20travel%20time.*) <br>  
Earth to Moon = 2,560 ms / 2 <br>  
Earth to Mars = 20 minutes on average (from reference above)

Output:

title: Assignment 1 subtitle: Computer performance, reliability, and scalability calculation author: Jake Meyer

**1.2**

**a. Data Sizes**

| **Data Item** | **Size per Item** |
| --- | --- |
| 128 character message. | 128 Bytes |
| 1024x768 PNG image | 3.15 MB |
| 1024x768 RAW image | 1.57 MB |
| HD (1080p) HEVC Video (15 minutes) | 224,004 MB |
| HD (1080p) Uncompressed Video (15 minutes) | 167,962 MB |
| 4K UHD HEVC Video (15 minutes) | 5,062.50 MB |
| 4k UHD Uncompressed Video (15 minutes) | 160,180.66 MB |
| Human Genome (Uncompressed) | ~200 GB |

Assumptions:

1. All videos are 30 frames per second.
2. HEVC stands for High Efficiency Video Coding
3. See the Wikipedia article on display resolution for information  
   on HD (1080p) and 4K UHD resolutions.

Calculations:  
1 byte = 1 letter in computer memory therefore 128 characters\* 1 byte = 128 Bytes  
[Image Size Calculator](https://toolstud.io/photo/filesize.php?imagewidth=1024&imageheight=768)  
PNG Image File Uncompressed 4x8bit RGBA  
Raw Image FIle Uncompressed 16bit monochrome  
[Video Size Calculator](https://www.omnicalculator.com/other/video-size)  
HD (1080p) Cineon 1080 RGB, 15 minute video, 30 frames per second  
HD (1080p) Uncompressed 1080 8-bit, 15 minute video, 30 frames per second  
[4K Video Size Calculator](https://www.videoproc.com/edit-4k-video/video-size-calculator.htm)  
4K UHD HEVC Video Resolution 3840 x 2160 from Wikipedia, 15 minutes, 30 frames per second  
4K UHD HEVC Uncompressed Resolution 1920 x 1080 pixels, 15 minutes, 30 frames per second  
[Human Genome Reference](https://medium.com/precision-medicine/how-big-is-the-human-genome-e90caa3409b0)

**b. Scaling**

|  | **Size** | **# HD** |
| --- | --- | --- |
| Daily Twitter Tweets (Uncompressed) | 178.81 GB | 1 |
| Daily Twitter Tweets (Snappy Compressed) | 3,039.83 GB | 1 |
| Daily Instagram Photos | 692,138.67 GB | 676 |
| Daily YouTube Videos | 78,751,406.25 GB | 76,906 |
| Yearly Twitter Tweets (Uncompressed) | 65,267.08 GB | 64 |
| Yearly Twitter Tweets (Snappy Compressed) | 1,109,540.46 GB | 1,084 |
| Yearly Instagram Photos | 252,630,859.38 GB | 246,710 |
| Yearly YouTube Videos | 28,744,263,281.25 GB | 28,070,570 |

Assumptions:

1. Using the estimates for data sizes in part a.
2. Using 10 TB Hard Drives and storing the data using the Hadoop Distributed File System (HDFS). 3x the amount of storage required.
3. 500 million tweets sent each day. Each tweet is 128 characters. Snappy compression ratio assumed 1:1.7.
4. 100 million videos and photos are uploaded to Instagram daily. 75% of those items are 1024x768PNG photos.
5. Youtube estimates 500 hours of video uploaded every minute. All videos are HD quality encoded using HEVC at 30 fps.

Calculations:  
Method used for calculations was to find the size initially for each request.  
Ensure the data size is multiplied by 3 since using HDFS.  
Use [Data Size Calculator](https://www.calculator.com/calculate/data-size/) to understand amount in Terabytes.  
Daily Twitter Tweets Size = 128 bytes \* 500 million tweets \* 3  
Daily Twitter Tweets Compressed Size = 128 bytes \* 1.7 Compression Ratio \* 500 million tweets \* 3  
Daily Instagram Photos Size = 3.15 MB \* 100,000,000 photos \* .75 \* 3  
Daily Youtube Videos Size = 224,004 MB \* 4 \* 500 hours of video \* 60 minutes \* 3  
Yearly Twitter Tweets Size = 128 bytes \* 500 million tweets \* 365 days \* 3  
Yearly Twitter Tweets Compressed Size = 128 bytes \* 1.7 Compression Ratio \* 500 million tweets \* 365 days \* 3  
Yearly Instagram Photos Size = 3.15 MB \* 100,000,000 photos \* .75 \* 365 days \* 3  
Yearly Youtube Videos Size = 224,004 MB \* 4 \* 500 hours of video \* 60 minutes \* 365 days\* 3

**c. Reliability**

|  | **# HD** | **# Failures** |
| --- | --- | --- |
| Twitter Tweets (Uncompressed) | 64 | 1 |
| Twitter Tweets (Snappy Compressed) | 1,084 | 15 |
| Instagram Photos | 246,710 | 3,380 |
| YouTube Videos | 28,080,570 | 384,704 |

Assumptions:

1. Use the yearly estimates from part b.

Calculations:  
[According to Backblaze hard drive statistics](https://www.backblaze.com/b2/hard-drive-test-data.html),  
annual failure rate is 1.37%. Calculations will be:  
Annual Number of Hard Drive Failures = Number of HD \* 0.0137  
Rounded to the nearest whole hard drive.

**d. Latency**

|  | **One Way Latency** |
| --- | --- |
| Los Angeles to Amsterdam | 70.36 ms |
| Low Earth Orbit Satellite | 100 ms |
| Geostationary Satellite | 140 ms |
| Earth to the Moon | 1,280 ms |
| Earth to Mars | 20 minutes |

Calculations:  
[Global Ping Statistics](https://wondernetwork.com/pings) used for latency from Los Angeles to Amsterdam  
Los Angeles to Amsterdam = 140.72 ms / 2  
[Low to Medium Earth Orbit Satellite](https://www.satellitetoday.com/telecom/2009/09/01/minimizing-latency-in-satellite-networks/)  
Low to Medium Earth Orbit Satellite = 100 ms taken from the article.  
[Geostationary Satellite Latency](https://www.satsig.net/latency.htm)  
Geostationary Satellite = 280 ms / 2  
[Earth to Moon and Earth to Mars](https://short-informer.com/what-is-the-latency-between-earth-and-moon/#:%7E:text=What%20is%20the%20latency%20between%20earth%20and%20moon%3F,to%20about%205.8%20milliseconds%20of%20wave%20travel%20time.)  
Earth to Moon = 2,560 ms / 2  
Earth to Mars = 20 minutes on average (from reference above)